Lecture 12: Kernel Methods

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

- Midterm on Thu (11/3) 10:00am-11:50am open notes/book/calculator
- You can attend the exam in 3 ways
  - online & zoom in (recommended)
  - in-person with your laptop (recommended)
  - request in-person paper exam

Midterm question types

Q1.3 ID3
2 Points

Which of the following statement(s) about ID3 algorithm are correct? Select all of them.

- [ ] The ID3 algorithm always finds the optimal decision tree, i.e., the decision treewith the minimal depth that can classify all training instances.
- [ ] The ID3 algorithm can be only used in binary classification problems.
- [ ] ID3 algorithm can be used to find a non-linear classifier.
- [ ] Decision trees can be implemented as a set of if-then-else statements.

Q1.1 Spam Filter Experiment
3 Points

Z is a summer intern working on spam classification in your company. The dataset consists of 10 million non-spam emails (class 0) and 10 thousand spam emails (class 1). Z considers the following steps of conducting experiments:

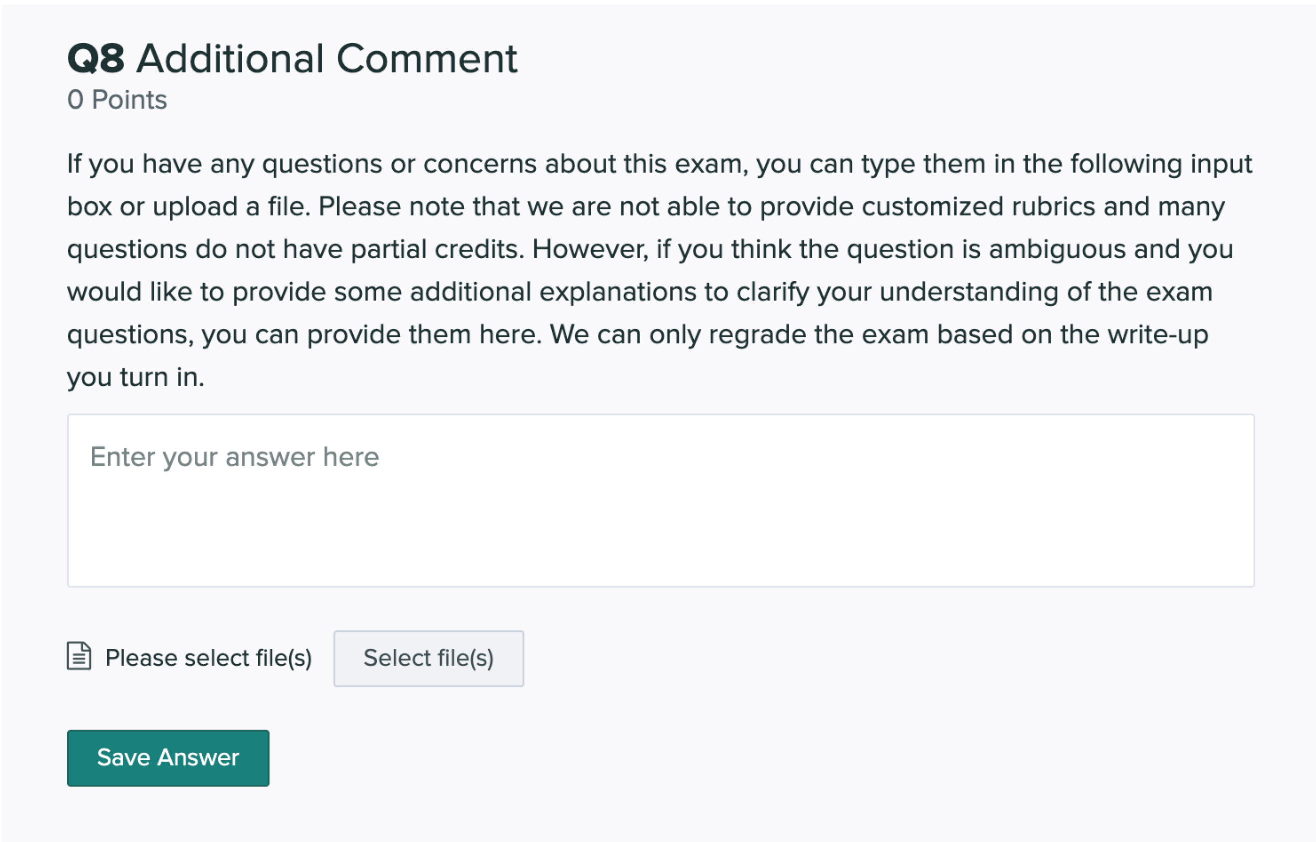
- Step 1: Shuffle the dataset and split it into the train, validation, and test sets.
- Step 2: Train logistic regression models on the train set with different hyper-parameters.
- Step 3: Identify the best hyper-parameter using the validation set and report the results on the test set in accuracy.

Do you agree with the above experimental setup?
If No, what is the major issue? Provide your suggestions in one or two sentences.

Enter your answer here

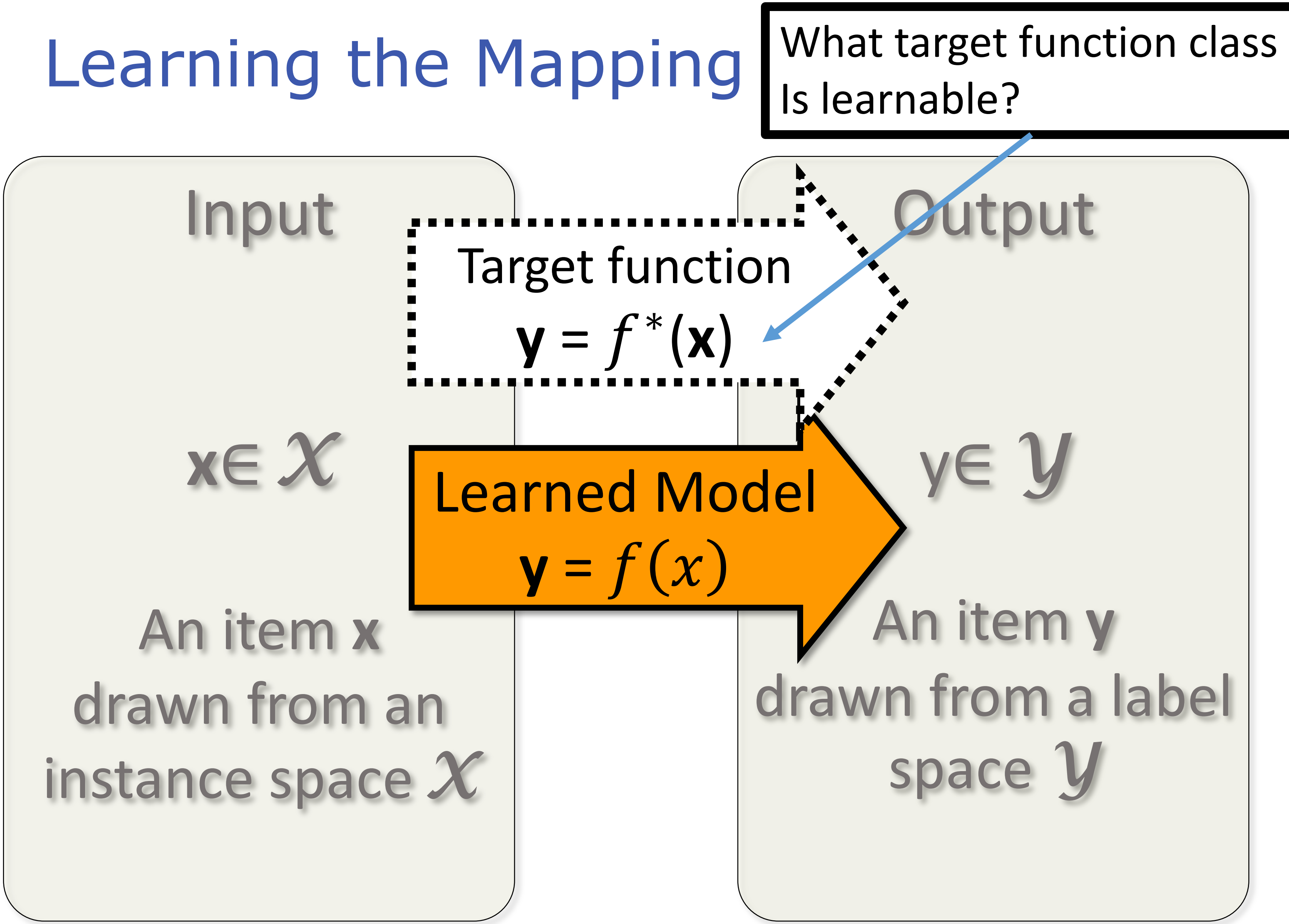
Midterm

- You can upload additional comments or additional explanation of your answers



Learning Theory

Learning the Mapping



Learning Monotone Conjunctions

- Hypothesis class:

$f = x_1?$ $\quad f = x_2?$ $\quad f = x_1 \wedge x_2 \wedge x_3\ ?$

....

$f = x_1 \wedge x_2?$ $\quad f = x_2 \wedge x_3\ ?$

- Target function in the hindsight

$$f = x_2 \wedge x_3$$

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a monotone conjunctive concept with n-dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

PAC Learnability

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is PAC learnable by L using H if for all $f \in C$, for all distribution D over X, and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D, the algorithm L produces, with probability at least (1- δ), a hypothesis h ∈ H that has error at most ε, where m is *polynomial* in 1/ ε, 1/ δ, n and size(H)

example: conjunction: $m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$

efficiently learnability

- The concept class C is *efficiently learnable* if L can produce the hypothesis in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and size(H)

PAC Learnability

- We impose two limitations
- Polynomial *sample complexity* (information theoretic constraint)
  - Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
- Polynomial *time complexity* (computational complexity)
  - Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

Will not be in exam

A general result (proof are skipped)

Let H be any hypothesis space.

With probability 1 -δ a hypothesis h → H that is consistent with a training set of size m will have an error < ϵ on future examples if

$$m > \boxed{\frac{1}{\epsilon}}\left(\boxed{\ln(|H|)} + \boxed{\ln\frac{1}{\delta}}\right)$$

1. Expecting lower error increases sample complexity (i.e more examples needed for the guarantee)

2. If we have a larger hypothesis space, then we will make learning harder (i.e higher sample complexity)

3. If we want a higher confidence in the classifier we will produce, sample complexity will be higher.

Will not be in exam

Example Disjunction

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is consistent with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

Size of hypothesis class for disjunction class $|H| = 3^n$, so a sufficient number of example to learn the disjunction concept is

$$\mathrm{m} > \frac{1}{\epsilon}\left(n \ln 3 + \ln\frac{1}{\delta}\right)$$

$\delta = \epsilon = 0.05, n = 10 \quad \Rightarrow m > 280$
$\delta = 0.01\ \epsilon = 0.05, n = 10 \quad \Rightarrow m > 312$

$\delta = \epsilon = 0.01, n = 10 \Rightarrow$ m > 1,625
$\delta = \epsilon = 0.01, n = 50 \Rightarrow$ m > 5,954

Will not be in exam

Example Arbitrary Boolean Function

Let H be any hypothesis space.

With probability 1 -δ a hypothesis h → H that is consistent with a training set of size m will have an error < ϵ on future examples if $m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$

Size of hypothesis class for Boolean functions is $|H| = 2^{2^n}$, so a sufficient number of example to learn the Boolean function concept is

$$\mathrm{m} > \frac{1}{\epsilon}\left(2^n \ln 2 + \ln\frac{1}{\delta}\right)$$

$\delta = \epsilon = 0.05, n = 10 \quad \Rightarrow m > 14{,}256$

$\delta = \epsilon = 0.05, n = 50 \quad \Rightarrow m > 1.5\times 10^{16}$

Will not be in the exam

How about real value functions

- VC(H) quantifies the complexity of the hypothesis space
  - E.g., VC(linear function) < VC(polynomial function)
  - E.g., VC(function w/ large margin) < VC(function w/ small margin)

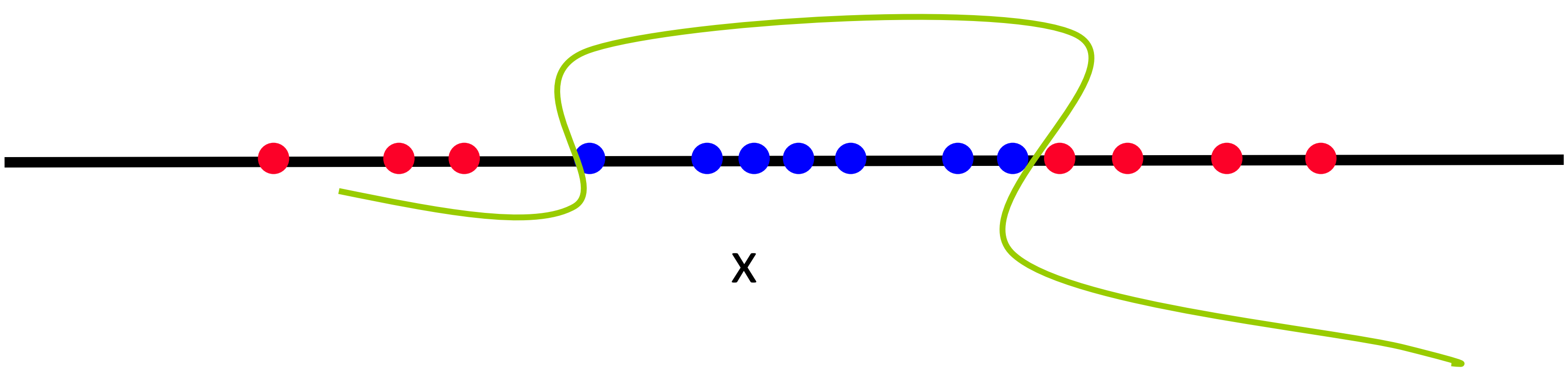
$$err_D(h) \leq err_S(h) + \sqrt{\frac{VC(H)\left(\ln \frac{2m}{VC(H)} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

error in test time error in training

Kernel and Kernel methods

Functions Can be Made Linear

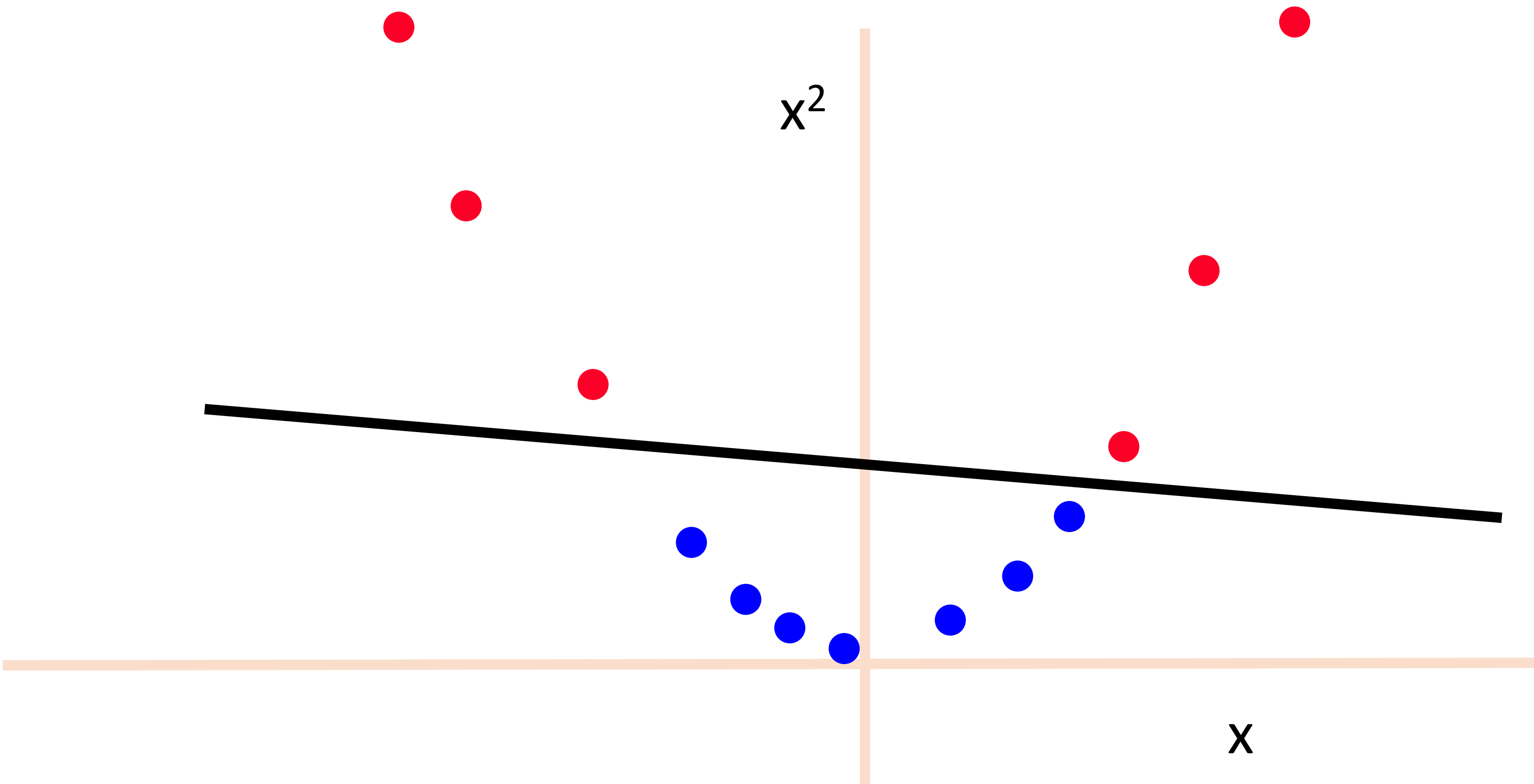
- Data are not linearly separable in one dimension
- Not separable if you insist on using a specific class of functions



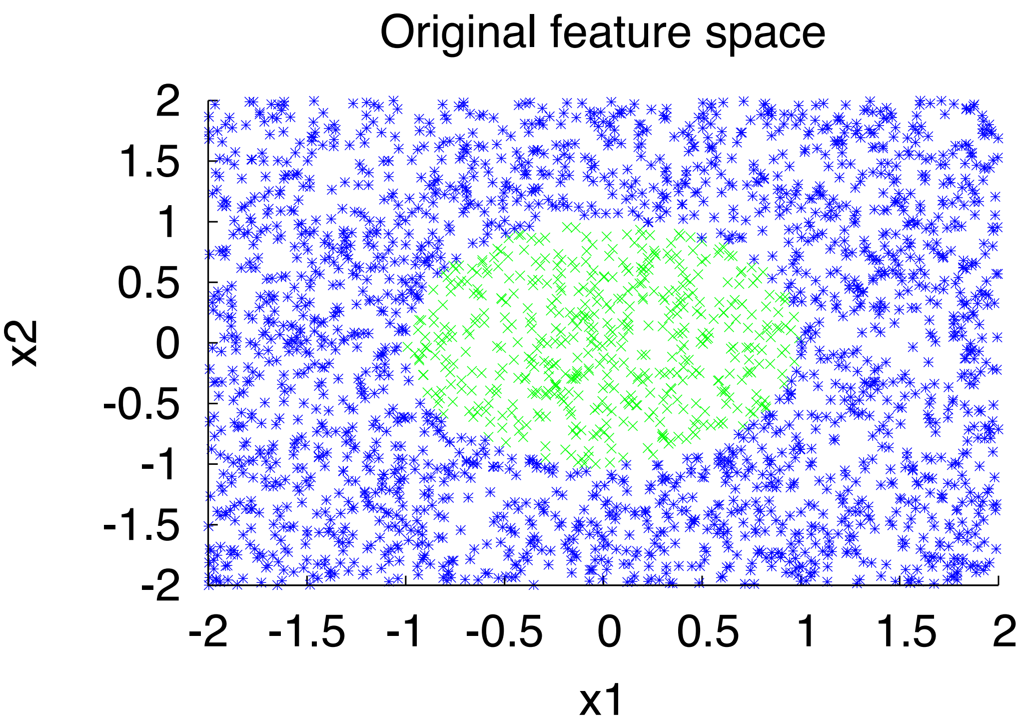
Can we do some mapping to make it linear spreadable?

Blown Up Feature Space

- Data are separable in <x, x^2> space

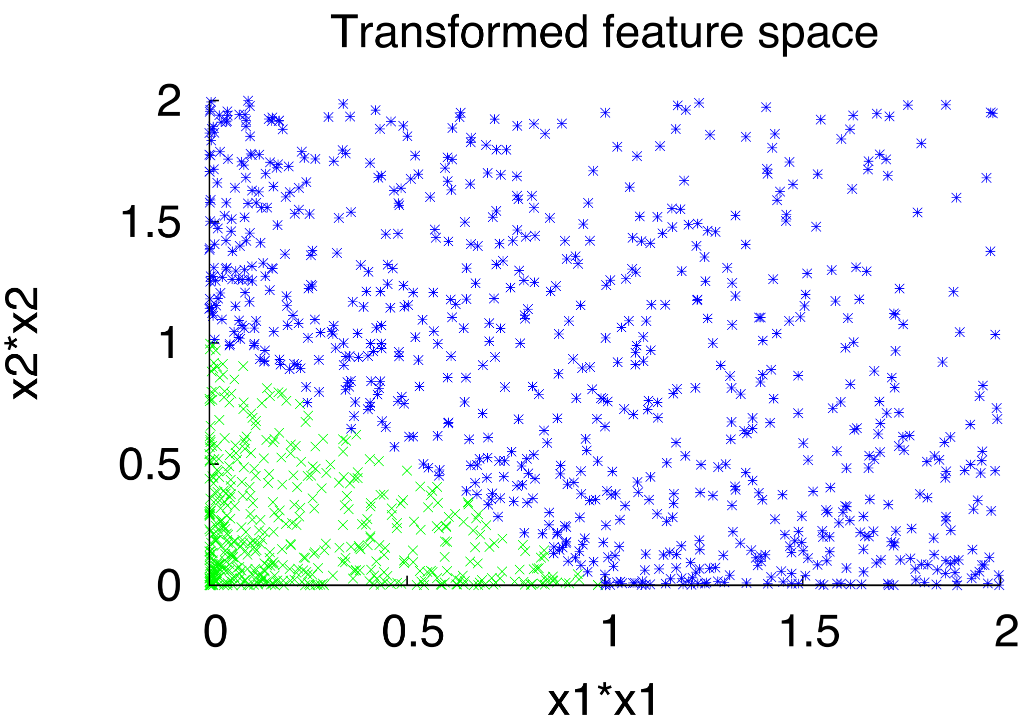


2D example



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Making data linearly separable



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Transform data: $\mathbf{x} = (x_1, x_2) \Rightarrow \phi(x) = (x_1^2, x_2^2)$

$f(\phi(x)) = 1$ iff $\phi(x)_1 + \phi(x)_2 \leq 1$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y(\boldsymbol{w}^\top \boldsymbol{x}) \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
5.
6. Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \boldsymbol{0} \in \mathbb{R}^{2n}$

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

    if $y\, \boldsymbol{w}^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq \boldsymbol{0}$

        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$

Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \operatorname{sgn}(\boldsymbol{w}^{\mathsf{T}} \begin{bmatrix} x \\ x^2 \end{bmatrix})$

The Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\mathrm{w} \leftarrow \mathbf{0}$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\, \phi(x)$
5.
6. Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \phi(x))$

The Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\mathrm{w} \leftarrow \mathbf{0}$

What if our mapping function is more complex?
E.g., mapping data to infinite # dimensions

Ans: it's okay if we can compute $\phi(x_i)^T \phi(x_j)$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \phi(x))$

$$\boxed{w^T \phi(x) = \sum_i \alpha_i \, y_i \phi(x_i)^T \phi(x)}$$

Infinite dimensions -> anything is linearly separable

Dual Representation

$$\text{if } y(\boldsymbol{w}^\top \boldsymbol{x}) \leq \boldsymbol{0}$$
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$$

- Let w be an initial weight vector for perceptron. Let $(x_1$,+), $(x_2$,+), $(x_3$,-), $(x_4$,-) be examples and assume mistakes are made on x_1, x_2 and x_4.
- What is the resulting weight vector?

$$w = w + x_1 + x_2 - x_4$$

- In general, the weight vector w can be written as a linear combination of examples:

$$w = \sum_{1..m} \alpha_i\, y_i x_i$$

- Where α_i is the the number of mistakes made on x_i.
- What will be the prediction on x

With the right mapping, the inner product phi(x_j)^T phi(x_i) can efficiently be computed. So, this algorithm will be O(m^2) where m is the size of the dataset.

The Dual Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \boldsymbol{R}^m$
2. For $(\boldsymbol{x}_i, y_i)$ in $\mathcal{D}$:
3. if $y_i \sum_j \alpha_j y_j \phi(x_i)^T \phi(x_j) \leq \mathbf{0}$
4. $\alpha_i \leftarrow \alpha_i + 1$
5. Return $\boldsymbol{\alpha}$
6.

$\boxed{\mathrm{w} \leftarrow \mathbf{0}}$

$\boxed{y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}}$

plug in w from below into this equation.

$\boxed{\boldsymbol{w} \leftarrow \boldsymbol{w} + y\, \phi(x)}$

If dimensionality of features n is too high, then we want to use this algorithm since our parameter alpha has dimension m << n.

alpha's dimension is |D| = m
w's dimension is the number of features per data point = n

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \phi(x)) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x)$

$$\boxed{w = \sum_{1..m} \alpha_i \, y_i \mathrm{phi}(\mathrm{x}_{i)}}$$

Predicting with linear classifiers

- Prediction = $sgn(\mathbf{w}^T\mathbf{x})$ and $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$
- That is, we just showed that

$$\mathbf{w}^T\mathbf{x} = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

  - Prediction can be done by computing dot products between training examples and the new example **x**

- This is true if we map examples with $\phi(x)$

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Relation to K-NN

- Linear model:

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

- K-NN --- votes are weighted by the distance between the neighbor and x

$$\arg max_y \sum_{x_i \in neighbor(x); y_i = y} \phi(x_i)^T \phi(x)$$

Dot product is negatively correlated to distance
Distance(x, y) = x^2 + y^2 - 2 * x . y

Dot products in high dimensional spaces

If $\phi(x)$ maps x to a high-dimensional space

We define $K(x, z)$ is the inner product of $\phi(x)$ and $\phi(z)$

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Dot products in high dimensional spaces

If $\phi(x)$ maps x to a high-dimensional space

We define $K(x, z)$ is the inner product of $\phi(x)$ and $\phi(z)$

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction is

$$sgn(\mathbf{w}^T \phi(\mathbf{x})) = sgn\left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})\right)$$

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Dot products in high dimensional spaces

If $\phi(x)$ maps x to a high-dimensional space

We define $K(x, z)$ is the inner product of $\phi(x)$ and $\phi(z)$

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction is

$$sgn(\mathbf{w}^T \phi(\mathbf{x})) = sgn\left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})\right)$$

If we can compute of $K(x_i, x)$ without explicitly writing the blown up representation, then we will have a computational advantage.

Example

❖ Assume the mapping:

Let $x \in R^D$

Computing $\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})$ is $\mathrm{O}(D^2)$.

$$\phi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_D \\ {x_1}^2 \\ x_1 x_2 \\ \vdots \\ x_1 x_D \\ x_2 x_1 \\ {x_2}^2 \\ \vdots \\ x_2 x_D \\ \vdots \\ x_D x_1 \\ \vdots \\ {x_D}^2 \end{pmatrix}$$

For example, D=1000

Dimension of $\phi(D)$ is $1 + 1000 + 1000^2$

Inner product can be computed efficiently

- However,

$$\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^2$$

Don't need to compute phi(x) to get the inner product. If D is the original dimension of x, this means we can compute the inner product in O(D).

- In fact, it can be computed in

$$O(D)$$

Just an exmaple, but this relation holds for all other dimensions.

Example: polynomial kernel

Let us examine more closely the inner products $\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$ for a pair of data points $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$.

Polynomial-based nonlinear basis functions consider the following $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{\phi} : \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) &= x_{m1}^2 x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2 x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2 \end{aligned}$$

Namely, the inner product can be computed by a function $(\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2$ defined in terms of the original features, *without computing $\boldsymbol{\phi}(\cdot)$.*

The Kernel Trick

Suppose we wish to compute

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Here ϕ maps **x** and **z** to a high dimensional space

The Kernel Trick: Save time/space by computing the value of K(**x**, **z**) by performing operations in the original space (without a feature transformation!)

Kernel functions

- A kernel function $k(\cdot,\cdot)$ satisfies the following properties
- For any x_m, x_n

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for some function $\phi(\cdot)$

Example: $(\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$ is a kernel, because it is the linear product of the following mapping

$$\boldsymbol{\phi} : \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Exercise

- Let $x \in R^2$, show $\left(4 + 9x_i^T x_j\right)^2$ is a valid kernel.

Zoo of Kernel

Linear Kernel: $K(x, y) = x^T y$

Polynomial Kernel: $K(x, y) = (x^\mathsf{T} y + c)^d$

RBF Kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$

The proof will not be in exam

RBF Kernel maps data into an infinite-dimension space

$$
\begin{aligned}
\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) &= \exp\left(\frac{2}{2}\mathbf{x}^\top\mathbf{x}' - \frac{1}{2}\|\mathbf{x}\|^2 - \frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \exp(\mathbf{x}^\top\mathbf{x}')\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\sum_{n_1+n_2+\cdots+n_k=j}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)\frac{{x'}_1^{n_1}\cdots {x'}_k^{n_k}}{\sqrt{n_1!\cdots n_k!}} \\
&= \langle\varphi(\mathbf{x}),\varphi(\mathbf{x}')\rangle
\end{aligned}
$$

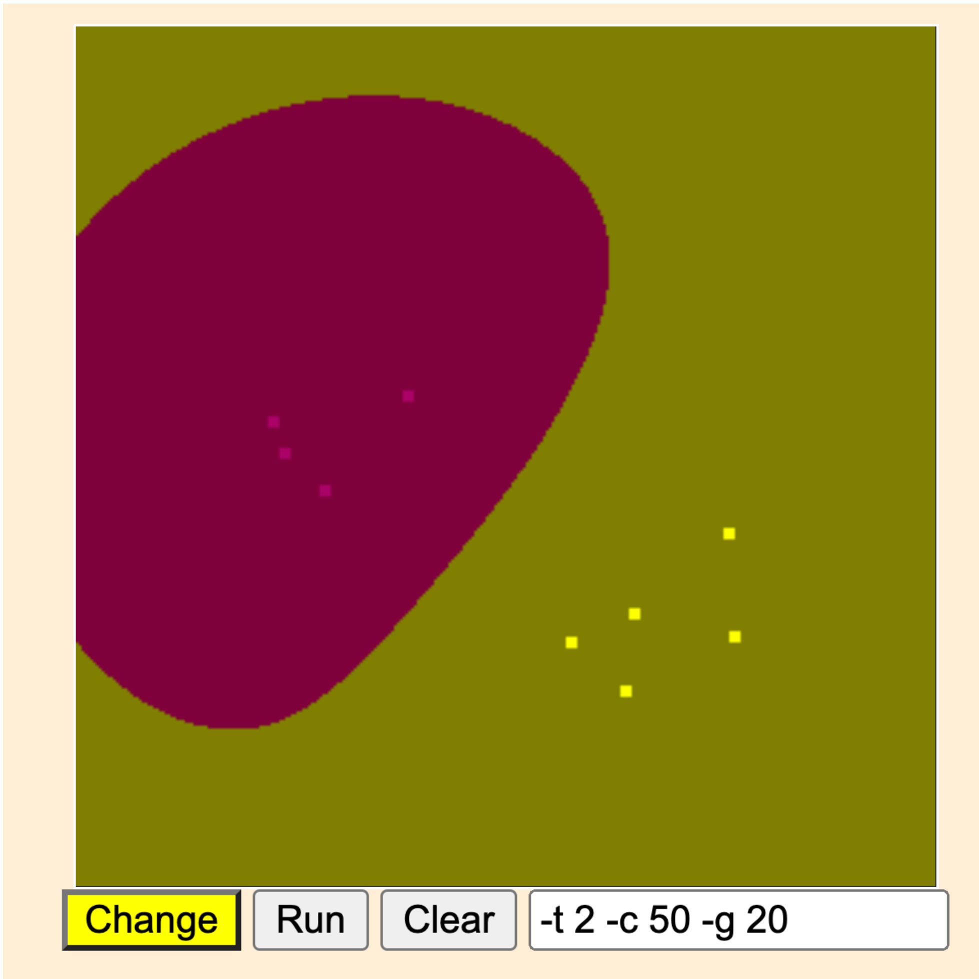
$$
\varphi(\mathbf{x}) = \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\left(a_{l_0}^{(0)}, a_1^{(1)}, \ldots, a_{l_1}^{(1)}, \ldots, a_1^{(j)}, \ldots, a_{l_j}^{(j)}, \ldots\right)
$$

where $l_j = \binom{k+j-1}{j}$,

$$
a_l^{(j)} = \frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}} \quad \Big| \quad n_1+n_2+\cdots+n_k = j \wedge 1 \le l \le l_j
$$

Demo – SVM (will be taught later)

- https://www.csie.ntu.edu.tw/~cjlin/libsvm/



The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \boldsymbol{R}^{2n}$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y\, \boldsymbol{w}^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$
5.
6.

Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \begin{bmatrix} x \\ x^2 \end{bmatrix})$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

1. Initialize $\mathrm{w} \leftarrow \mathbf{0}$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\, \phi(x)$
5.
6. Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^{\mathsf{T}} \phi(x))$

$$\boxed{w^T \phi(x) = \sum_i \alpha_i\, y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)}$$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$

```
Initialize α ← 0 ∈ R^m
For (x_i, y_i) in D:
    if y_i Σ_j α_j y_j K(x_i, x_j) ≤ 0
        α_i ← α_i + 1
Return w
```

$\mathrm{w} \leftarrow \mathbf{0}$

$y\, \boldsymbol{w}^T \phi(x) \leq \mathbf{0}$

$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\phi(x)$

Prediction: $y^{\text{test}} \leftarrow \mathrm{sgn}(\boldsymbol{w}^{\mathsf{T}} \phi(x)) = \sum_i \alpha_i y_i K(x_i, x)$

$$\boxed{w^T \phi(x) = \sum_i \alpha_i\, y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)}$$